CI/CD with Zuul and Kubernetes

Joshua Watt, Garmin

Yocto Project Summit, 2021.11

About Me

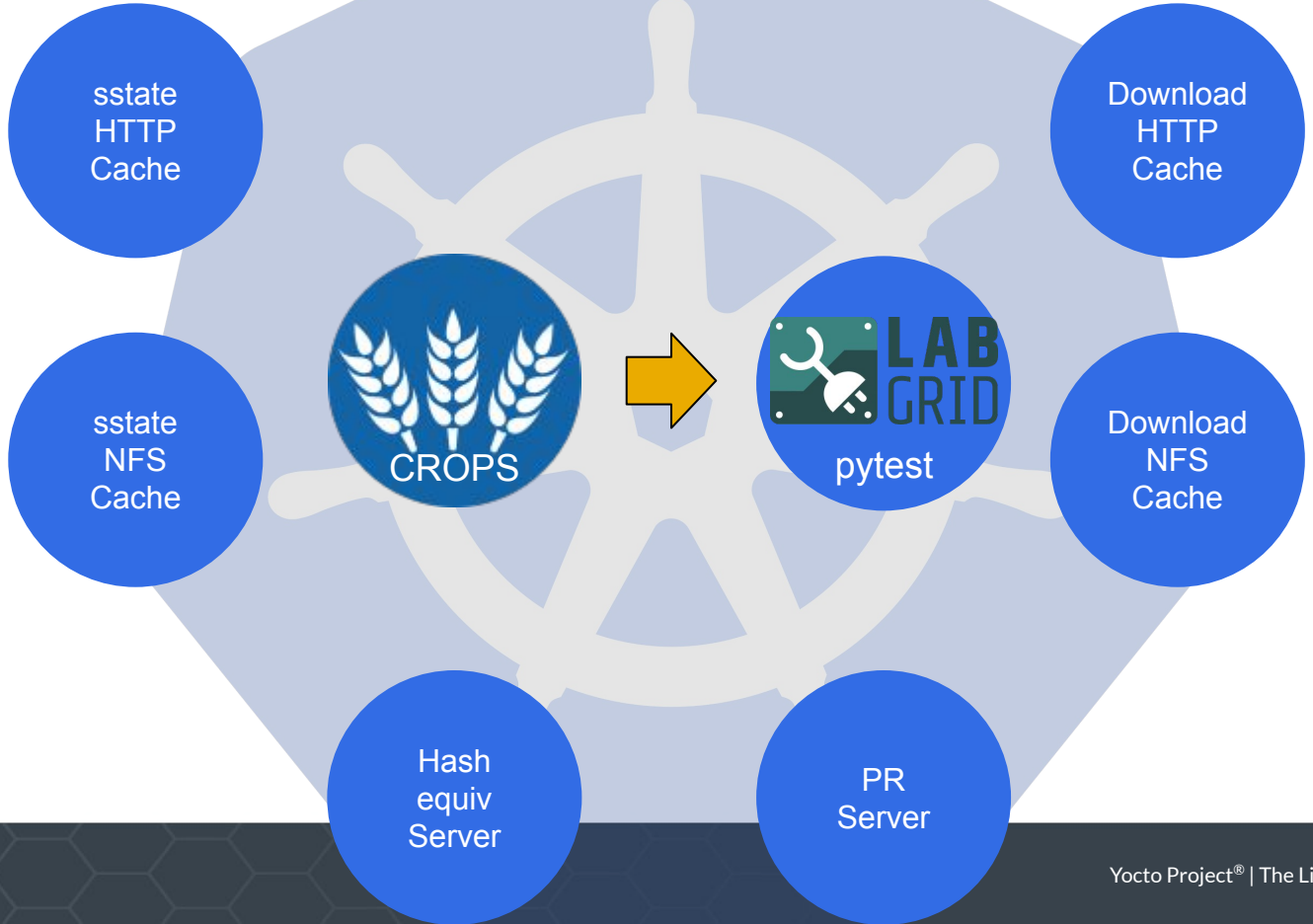
- Worked at Garmin since 2009
- Using OpenEmbedded & Yocto Project since 2016
- Joshua.Watt@garmin.com
- JPEWhacker@gmail.com
- IRC (OFTC or libera): JPEW
- Twitter: @JPEW_dev
- LinkedIn: joshua-watt-dev

Introduction & Background

Overview

This is a continuation of my 2021 ELC presentation:

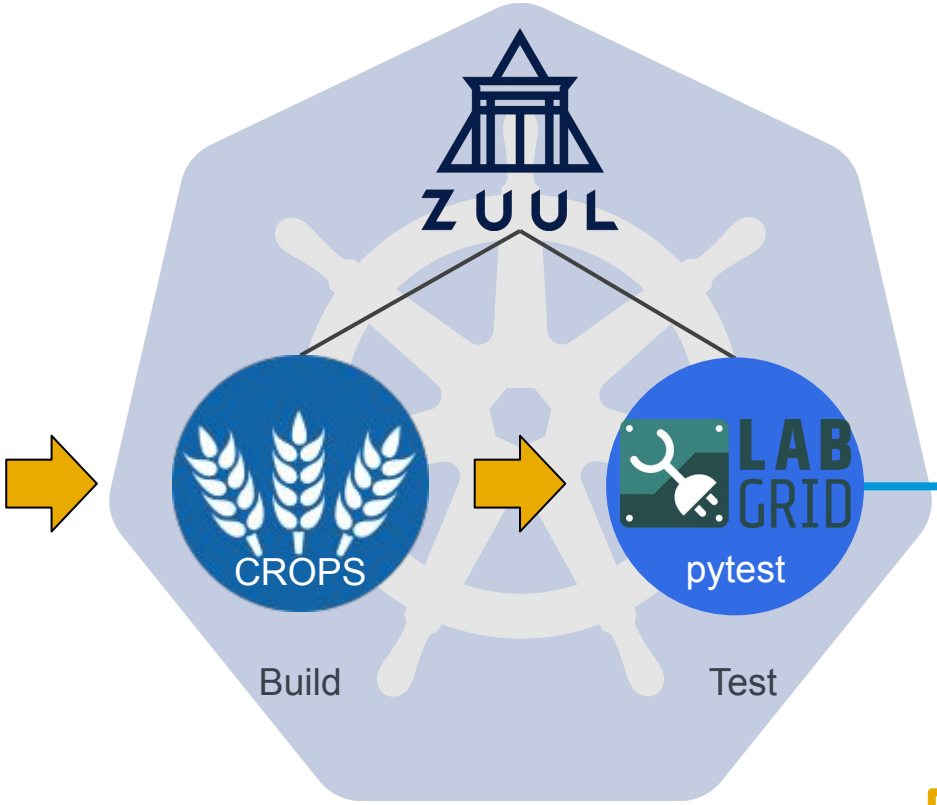
https://www.youtube.com/watch?v=GAkpFUmoMT0


sstate
HTTP
Cache
sstate
NFS
Cache
CROPS
LAB
GRID
pytest
Download
HTTP
Cache
Download
NFS
Cache
Hash
equiv
Server
PR
Server

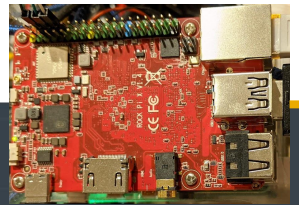
```
private static final String[] natoCalls = {
    "Alpha", "Bravo", "Charlie", "Delta", "Echo",
    "Foxtrot", "Golf", "Hotel", "India", "Juliet",
    "Kilo", "Lima", "Mike", "November", "Oscar",
    "Papa", "Quebec", "Romeo", "Sierra", "Tango",
    "Uniform", "Victor", "Whiskey", "X-ray", "Yankee", "Zulu"
};

/**
 * @brief Returns the NATO call for a letter, eg. F = Foxtrot.
 * @param letter A character in range a-z, A-Z or 0-9
 */
public static String getNATOCall(char letter){
    if(letter >= 'a' && letter <= 'z'){
        return natoCalls[letter-'a'];
    }
    if(letter >= 'A' && letter <= 'Z'){
        return natoCalls[letter-'A'];
    }
    return null;
}
```

Code (meta-phosh)



SSH

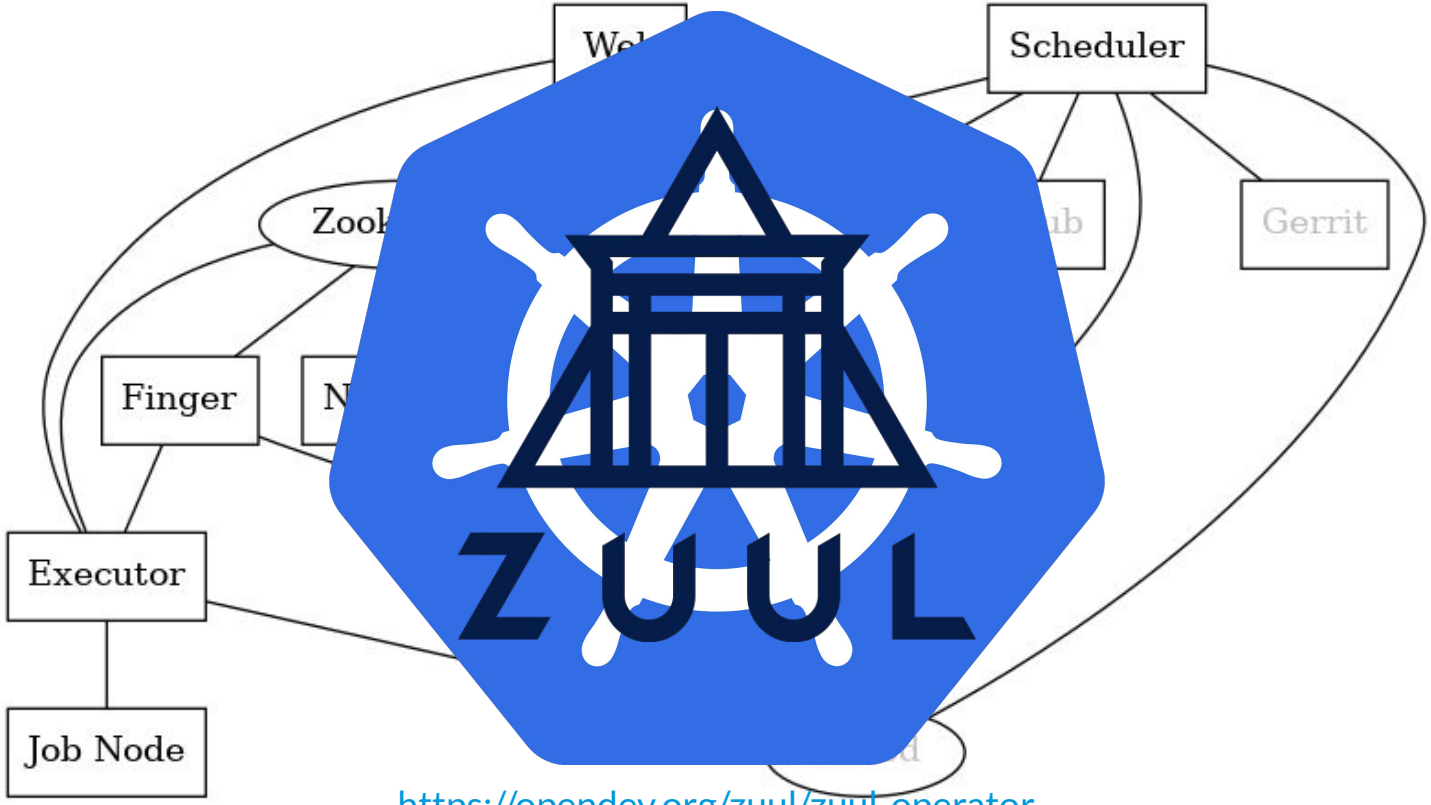


What Is Zuul?

Zuul CI Project

- Open Source Project
- Primarily developed by OpenStack
- Designed for HA and Scalability
- Highly flexible configuration
- https://zuul-ci.org/

Zuul Components



https://opendev.org/zuul/zuul-operator

Supported Zuul Platforms

Source Drivers

- Gerrit
- GitHub
- GitLab
- Timer

Node Backends

- Kubernetes Pods
- AWS/GCP/Azure
- OpenStack
- Static Nodes

Zuul Configuration: Pipelines

- When to build (triggers)
- What to do when a build finishes
  - Merge changes
  - Comment
  - etc.
- YAML files
- Defined in a trusted configuration git repo
- https://gitlab.com/JPEWdev/zuul-config/-/blob/master/zuul.d/pipelines.yaml

Zuul Configuration: Projects

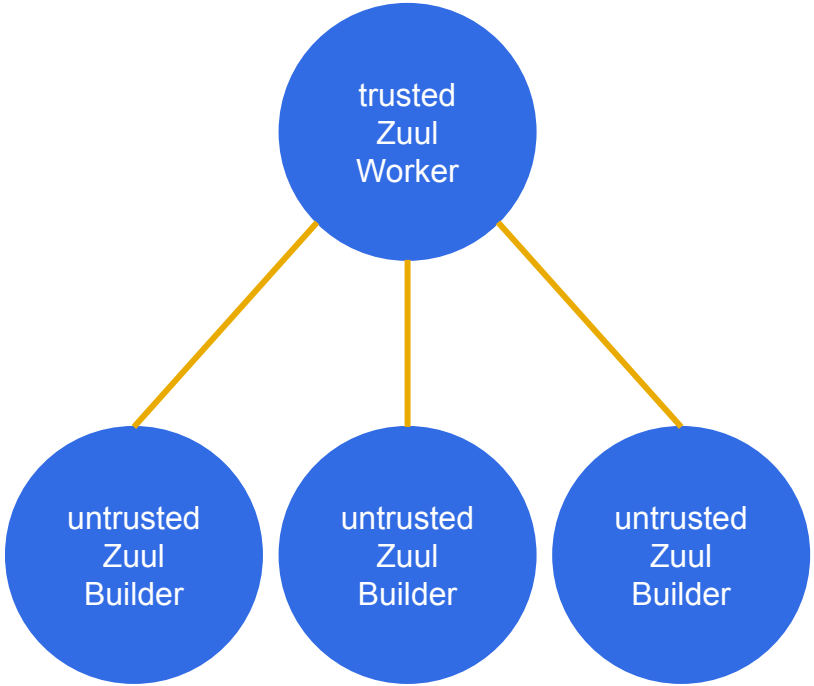
- **What to build (source code)**
- **YAML files**
  - Most commonly lives with the code to be built
  - Can also live in a trusted git repo

Zuul Configuration: Jobs

- **How to build**
- **YAML files**
- **Configuration can be easily shared**
  - Simple to create common "libraries" of functionality in a common git repo
  - https://gitlab.com/JPEWdev/yocto-zuul-jobs
- **Supports Inheritance**
- **Job actions are defined using Ansible**

Zuul Job Execution

- **Execution is divided between trusted worker node, and untrusted builders**
  - Allows limiting automated testing of unreviewed code



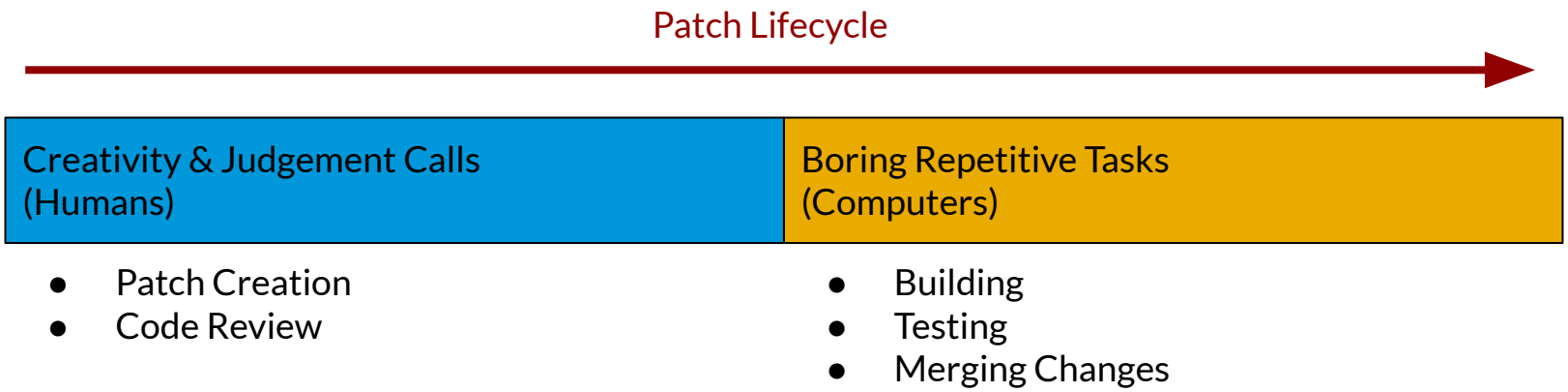
Cross-Project Dependencies

- Changes can be declared as dependencies across multiple projects
- Zuul will ensure dependencies are merged together to prevent breakage between repos

Project Gating

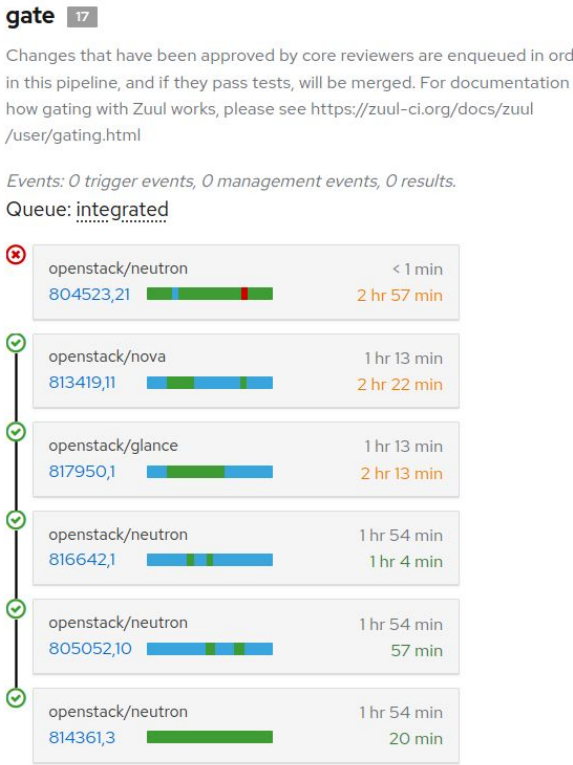
Every change is built and tested or it doesn't merge

https://gating.dev/



Speculative Execution

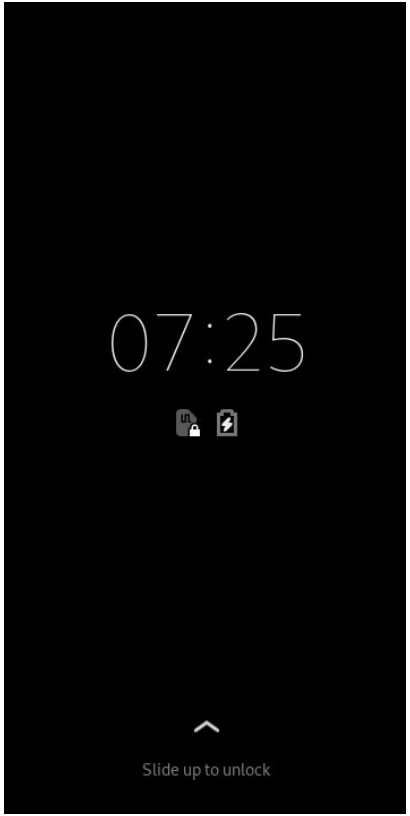
- Gating changes are "stacked" and tested in-situ
- Zuul can work ahead in the stack
  - Changes later in the stack can be built and tested before previous changes complete (assumes they will pass)
- Optimizes CI resource usage
- Further reduces requirement for developer interaction



Zuul and meta-phosh

What is meta-phosh?

- Layer for building GNOME based PHOSH shell
- https://github.com/JPEWdev/meta-phosh
- Tim Orling has an entire presentation on this later



meta-phosh Zuul Project

```
$ cat zuul.d/projects.yaml
- project:
    check:
      jobs:
      - phosh-qemux86-64
      - phosh-raspberrypi4-64
    gate:
      jobs:
      - phosh-qemux86-64
      - phosh-raspberrypi4-64
      - phosh-qemux86-64-test
      - phosh-raspberrypi4-64-test
    periodic-daily:
      jobs:
      - phosh-qemux86-64
      - phosh-raspberrypi4-64
      - phosh-qemux86-64-test
      - phosh-raspberrypi4-64-test
      - phosh-publish-daily
```

meta-phosh Pipelines

- **check**
  - Run everytime a pull-request is created/updated
  - Basic sanity checks
  - Currently builds images (will change this)
  - https://zuul.wattissoftware.com/t/wattissoftware-zuul/buildsets?project=JPEWdev%2Fmeta-phosh&branch=master&pipeline=check

meta-phosh Pipelines

- **gate**
  - Triggered by "gate" label in pull-request
  - Run after review when pull-requests is ready to be merged
  - Builds qemux86-64 & raspberrypi4-64 images
  - Tests images on Labgrid cluster
  - Merges changes
  - https://zuul.wattissoftware.com/t/wattissoftware-zuul/buildsets?project=JPEWdev%2Fmeta-phosh&branch=master&pipeline=gate

meta-phosh Pipelines

- **periodic-daily**
  - Runs every night
  - Builds qemux86-64 & raspberrypi4-64 images with latest upstream layers
  - Tests images on Labgrid cluster
  - Publishes images for download
  - https://zuul.wattissoftware.com/t/wattissoftware-zuul/buildsets?project=JPEWdev%2Fmeta-phosh&branch=master&pipeline=periodic-daily
  - http://downloads.wattissoftware.com/publish/

Limitations

Gating Trigger on GitHub

- **Using a label on a PR in GitHub is not ideal**
  - Works much better with GitHub Code Review approval…
  - Can't self approve a PR 😞
  - GitLab works better for this

Manually Triggering Jobs

- Zuul works great for external "triggers" (e.g. new Pull Request, tag, pushed commit, timer, etc.)
- Triggering a "manual" build at a specific commit can be tricky

Build Parameterization

- Zuul doesn't have a concept of build "parameters"
- Difficult to make on-the-fly customizations for "manual" builds

Future Direction

Build "world" and publish package feeds

- Weekly(?)
- Need a more robust hosting solution
- Also need PR server

Build using kas YAML files

- **Pros:**
  - Would test that kas files work for end users
  - Common ansible roles would be easy to share
- **Cons:**
  - Build kas without a container?
  - Loose cross-project tracking in Zuul
  - Dependent layers would be downloaded every build

Automatic Recipe Upgrades

- Use AUH or similar to automatically update recipes
- Test the upgrades on hardware using Labgrid
- This is a little trickier with GitHub PRs

Questions?

yocto PROJECT | THE LINUX FOUNDATION